

(19)日本国特許庁 (J P)

(12) 公 開 特 許 公 報 (A)

(11)特許出願公開番号

特開2000-298652

(P2000-298652A)

(43)公開日 平成12年10月24日(2000. 10. 24)

(51)Int.Cl. ⁷	識別記号	F I	テーマコード*(参考)
G 0 6 F 15/16	6 4 0	G 0 6 F 15/16	6 4 0 B 5 B 0 4 5
15/78	5 1 0	15/78	5 1 0 G 5 B 0 6 2
			5 1 0 A

審査請求 未請求 請求項の数7 O L (全 8 頁)

(21)出願番号 特願平11-106052

(22)出願日 平成11年4月14日(1999. 4. 14)

(71)出願人 000006013

三菱電機株式会社

東京都千代田区丸の内二丁目2番3号

(72)発明者 鈴木 和雅

東京都千代田区丸の内二丁目2番3号 三

菱電機株式会社内

(72)発明者 服部 孝

東京都千代田区丸の内二丁目2番3号 三

菱電機株式会社内

(74)代理人 100099461

弁理士 溝井 章司 (外2名)

Fターム(参考) 5B045 EE01 EE03 EE14 KK08

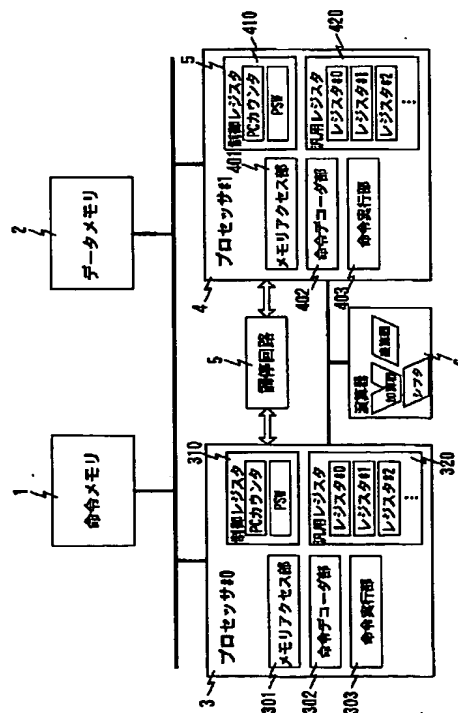
5B062 CC04 DD02 DD05 DD10

(54)【発明の名称】 マルチプロセッサ

(57)【要約】

【課題】 単体のプロセッサよりも複数のプロセスを効率良く実行できるようにし、従来のマルチプロセッサよりも面積を小さくする。

【解決手段】 複数のプロセッサ3、4と、複数のプロセッサから共有される命令メモリ1及びデータメモリ2と、複数のプロセッサから共有される演算器6を備え、更に命令メモリ1、データメモリ2、及び演算器6が同時期には1つのプロセッサからのみアクセスされるように調停する調停回路5を備える。



1

【特許請求の範囲】

【請求項 1】 複数のプロセッサと、前記複数のプロセッサから使用される共有演算器と、前記複数のプロセッサのうちのプロセッサが前記共有演算器を使用している間は他のプロセッサが前記共有演算器を使用できないように制御する調停回路を備えることを特徴とするマルチプロセッサ。

【請求項 2】 前記調停回路は、前記一のプロセッサの前記共有演算器の使用を許可する信号を前記一のプロセッサに送出するとともに、前記他のプロセッサの前記共有演算器の使用を拒絶する信号を前記他のプロセッサに送出することを特徴とする請求項 1 記載のマルチプロセッサ。

【請求項 3】 前記調停回路は、前記共有演算器を使用できる前記一のプロセッサと前記他のプロセッサとを一定時間ごとに切り替えることを特徴とする請求項 1 記載のマルチプロセッサ。

【請求項 4】 複数のプロセッサと、前記複数のプロセッサから使用される共有演算器と、前記複数のプロセッサから使用される共有メモリと、前記複数のプロセッサのうちのプロセッサが前記共有演算器を使用している間は他のプロセッサが前記共有演算器を使用できないように制御するとともに前記複数のプロセッサのうちのプロセッサが前記共有メモリを使用している間は他のプロセッサが前記共有メモリを使用できないように制御する調停回路を備えることを特徴とするマルチプロセッサ。

【請求項 5】 前記調停回路は、前記共有演算器の使用については前記一のプロセッサの前記共有演算器の使用を許可する信号を前記一のプロセッサに送出するとともに、前記他のプロセッサの前記共有演算器の使用を拒絶する信号を前記他のプロセッサに送出し、前記共有メモリの使用については前記一のプロセッサの前記共有メモリの使用を許可する信号を前記一のプロセッサに送出するとともに、前記他のプロセッサの前記共有メモリの使用を拒絶する信号を前記他のプロセッサに送出することを特徴とする請求項 4 記載のマルチプロセッサ。

【請求項 6】 前記調停回路は、前記共有演算器の使用については前記共有演算器を使用できる前記一のプロセッサと前記他のプロセッサとを一定時間ごとに切り替え、前記共有メモリの使用については前記共有メモリを使用できる前記一のプロセッサと前記他のプロセッサとを一定時間ごとに切り替えることにより制御することを特徴とする請求項 4 記載のマルチプロセッサ。

【請求項 7】 前記共有演算器と前記複数のプロセッサとが前記共有メモリと前記複数のプロセッサとの間の接続と別個に接続されていることを特徴とする請求項 4 記載のマルチプロセッサ。

【発明の詳細な説明】

【0001】

2

【発明の属する技術分野】この発明は、複数のプロセッサを実装したマルチプロセッサに関するものである。特に、単一の LSI 上に複数のプロセッサを実装した 1 チップマルチプロセッサにおいて、複数のプロセッサで回路を共有することにより LSI 面積を抑えつつ、複数のプロセスを効率よく実行することが可能なマルチプロセッサに関するものである。

【0002】

【従来の技術】単一のプロセッサからなるシングルプロセッサ上で複数の処理を実行する手段として代表的なものに、OS（基本ソフトウェア）をベースにおき、OS 上で各種アプリケーションを実行するというものがある。OS はあるプログラムの処理単位（以後、プロセスと呼ぶ）に分けて各アプリケーションを実行する。同時に 1 つのプロセッサ上で実行できるプロセスは 1 つであるため、ある時間毎に実行するプロセスを細かく切り替えることによって、あたかも複数のプロセスを同時に実行しているかのように見せている。

【0003】図 9 は、2 つのプロセスが時分割で実行されているときの、プロセッサ上で動作している処理の様子を示したもので、4001、4002 はそれぞれ異なる 2 つのプロセス プロセス A、プロセス B を示し、4003 は、2 つのプロセスの切り替え（以後、「プロセススイッチ」という）のために必要な処理を示している。プロセスを時分割で切り替える一般的な手段としては、タイマーからの割り込みを利用する方法がある。例えば、タイマーからの割り込み回数をカウントしておき、あらかじめ決められたあるカウント値に達したら、プロセススイッチを行うようにする。プロセススイッチでは、プロセッサの視点から見た場合には、切り替える前に実行していたプロセスに関するデータ（プログラムカウンタ（PC）値、スタックポインタアドレス、汎用レジスタ、プログラム状態語（PSW）中の必要な情報など）のメモリへの待避、切り替え先のプロセスに関するデータ（PC 値、スタックポインタ、汎用レジスタ、PSW 内の情報など）のデータメモリからのリストアが行われる。そのため、プロセススイッチが実行されると、プロセッサでは数十～数百の命令が実行されることになり、次のプロセスへの切り替えに時間を要し、高速処理が図れないという問題があった。

【0004】この問題を解決し、種々の処理を高速に実行する手法の 1 つとして、マルチプロセッサシステムがある。これは、必要とされる処理を、複数のプロセッサに分散して並列に処理することによって、高速化を計るものである。近年の LSI 集積技術は、このマルチプロセッサシステムを単一の LSI 上に実装することを可能にし、マルチプロセッサシステムを安価に実現することを可能にした。しかし、マルチプロセッサは、高速処理が図れる一方で、複数のプロセッサが各々命令メモリ等のハードウェア・リソースを有するため面積的に不利と

3

いう問題点がある。この問題を解決する手段として、例えば、特開昭 62-24356 で開示された 1 チップマイクロコンピュータがある。特開昭 62-24356 では、特に ROM、RAM を共用することにより面積的に有利にするとともに、各プロセッサでの ROM、RAM をアクセスするタイミングをずらすことにより効率的にプログラムを実行することが可能なマルチプロセッサシステムについて開示している。

【0005】

【発明が解決しようとする課題】 以上のように、従来の単一のプロセッサで複数のプロセスを実行する際には、プロセススイッチが発生してしまう。このため、ユーザからは実際には必要としない処理に対して、時間を消費してしまうことになり、高速処理が実現できないという問題がある。マルチプロセッサでは、各プロセッサで実行するプロセスを分けることにより、上記の問題を解決することが可能であるが、それぞれのプロセッサが独立してハードウェア・リソースを持つため、面積的に不利との問題点があった。この点につき、従来は ROM、RAM といったメモリを共有する手段を採用していたが、近年のプロセッサの省面積要求に対して、このメモリの共有のみでは十分とはいえず、更なる省面積化を実現するマルチプロセッサ技術が望まれていた。そこで、この発明は上記のような問題点を解決するためになされたもので、複数のプロセスが実行されるときにも効率よく実行することができ、かつ省面積なプロセッサを実現することを目的とする。

【0006】

【課題を解決するための手段】 この発明に係るマルチプロセッサは、複数のプロセッサと複数のプロセッサから共有されるメモリ、更には複数のプロセッサから共有される演算器を備えるものであり、メモリ、及び演算器は同時期には一つのプロセッサからのみアクセスされるように調停する手段を備えるものである。具体的には、以下の特徴を有する。

【0007】 命令コードを記憶する命令メモリと、被演算データや演算結果を記憶するデータメモリと、演算器を内部に含まない複数のプロセッサと、プロセッサからの制御データに従って演算を行う演算器とを持ち、前記命令メモリ、前記データメモリ、及び前記演算器は、複数の前記プロセッサによって共有され、それぞれ同時期には多くとも一つの前記プロセッサのみからアクセスされるようにする手段を持つことを特徴とする。

【0008】 一定の期間ごとに、命令メモリ、データメモリ、及び、演算器を使用することが可能なプロセッサを切り替える手段を持つことを特徴とする。

【0009】

【発明の実施の形態】 実施の形態 1. 図 1、図 2 はこの発明の実施の一形態を示す 1 チップマルチプロセッサの構成図である。説明を簡単にするため、ここではプロセ

4

ッサが 2 つの場合を例にとって説明する。図において、1 は命令コードを記憶する命令メモリ、2 は、被演算データや命令の実行結果などを記憶するデータメモリ、3 は命令コードに従って命令を実行するプロセッサ #0 であり、内部に演算器を含んでいない。4 はプロセッサ #1 であり、プロセッサ #0 (3) と同様に内部に演算器を含んでいない。5 は調停回路であり、命令メモリ 1、データメモリ 2 及び演算器 6 をどちらのプロセッサが使用するかを決定する。つまり、調停回路 5 は、共有メモリ (命令メモリ及びデータメモリ) のみならず、演算器 6 の使用についても調停する。6 は演算器であり、演算命令を実行する際に必要となる加算器や乗算器などを含む。また、演算器はプロセッサ #0 (3) 及びプロセッサ #1 (4) に共有されている。301、401 はメモリアクセス部であり、命令コードを命令メモリ 1 から読み出す (フェッチする) ときや、ロード/ストア命令において、レジスタとメモリ間でデータ転送を行うときに、命令メモリ 1 やデータメモリ 2 に対してアクセスを行う。302、402 は命令デコード部であり、読み出した (フェッチした) 命令コードをデコードし、適切な制御信号を命令実行部 303、403 や調停回路 5 に伝える。303、403 は命令実行部であり、命令デコード部 302 や調停回路 5 から送出された制御信号に従って、演算器 6 に制御信号や必要なデータを送出したり、演算器 6 が実行した演算結果をラッチして汎用レジスタ 320、420 に書込んだりする。また、310、410 は制御レジスタであり、現在実行中のプログラムの次に読み出す (フェッチする) 命令コードの命令メモリ 1 上のアドレスを保持するプログラムカウンタや、プロセッサの状態などの情報を保持するプログラムステータスワード等から構成される。320、420 は汎用レジスタであり、演算結果データや被演算データなどを一時的に保持するレジスタから構成される。図 1 では、命令メモリ 1、データメモリ 2、及び演算器 6 が共通のバスを介してプロセッサ #0 (3) とプロセッサ #1 (4) に接続している構成を示してしている。図 2 においては、演算器 6 は、命令メモリ 1、データメモリ 2 とは別のバスを介して各プロセッサに接続されている。図 1 の構成ではバスを共通にするため、バスの数を減少させることができるとのメリットがある。一方で、図 1 では、バスが共通するためデータの送受信が混み合うとのデメリットがあり、図 2 の構成は演算器 6 と共有メモリ (命令メモリ及びデータメモリ) で使用するバスを別に行っているのでデータ送受信が円滑に行えるとのメリットがある。これらの構成は、プロセッサの使用目的や使用条件等によって選択することができる。

【0010】 図 3 は、プロセッサ #0 (3) 及びプロセッサ #1 (4) における演算命令のパイプラインの構成を示したもので、4 段のパイプラインから構成されている。図 3 において、20 は命令を読み出す (フェッチす

5

る) 期間 (以下、「IF ステージ」という) であり、21 は、命令をデコードする期間 (以下「D ステージ」という) であり、22 は、デコードした命令を実行する期間 (以下「E ステージ」という) であり、23 は、演算した結果をレジスタに書き戻す期間 (以下、「W ステージ」という) である。次に動作について説明するが、プロセッサ #0 (3) もプロセッサ #1 (4) も基本的な動作は同じなので、特にことわりがない限り、プロセッサ #0 (3) に関して説明していく。IF ステージ 20 では、PC カウンタが示すアドレスにある命令コードをメモリアクセス部 301 を経由して命令メモリ 1 から読み出す。次に、D ステージ 21 では、読み出した命令コードを命令デコーダ部 302 にてデコードし、デコード結果を命令実行部 303 に送出する。このとき、実行する命令コードが演算命令の場合には、調停回路 5 に対して、次のステージで演算器 6 へのアクセスを行う旨のリクエスト信号を送出する。そして、E ステージ 22 では、命令デコーダ部 302 から出力された制御信号、及び、調停回路 5 からのアクセスを許可することを意味するアクノーリッジ信号に従って、演算器 6 に対して被演算データ及び演算の種別を示す制御コードを送出する。演算器 6 では、受け取ったデータ及び制御コードに従って演算を行った後、命令実行部 303 に演算結果を送出する。最後に、W ステージ 23 では、命令実行部 303 は、演算器 6 から受け取った演算結果を、汎用レジスタ 320 に書込む。調停回路 5 では、プロセッサ #0

(3)、プロセッサ #1 (4) が命令メモリ 1、データメモリ 2、演算器 6 の各リソースを使用するときの調停を行う。リソースを使用する側のプロセッサは、調停回路 5 に対して、使用する各リソースに応じたリクエスト信号を送出し、それを受け取った調停回路 5 は、使用の許可を意味するアクノーリッジ信号を、使用を許可するプロセッサに対して送出する。調停回路 5 では、各リソースに対して、先にリクエスト信号を出したプロセッサに対して、アクノーリッジ信号を送出するが、複数のプロセッサから同時期にリクエスト信号が送出されたときには、優先度に従ってアクノーリッジ信号を送出し、他方のプロセッサに対してはアクセスを拒絶する信号を送出する。この優先度の付け方に関しては、例えば、固定にしてもよいし、順番に各プロセッサ間で優先度が変わるようにしてもよい。

【0011】図 4 は、プロセッサ #0 (3)、プロセッサ #1 (4) で、それぞれ異なる一連の演算命令を実行したときのパイプラインの様子を示したものである。30、32、34 は、プロセッサ #0 (3) において、この順番に実行される一連の演算命令であり、31、33、35 は、プロセッサ #1 (4) において、この順番に実行される一連の演算命令である。プロセッサ #0 (3)、プロセッサ #1 (4) から同時期に命令メモリ 1 のリクエスト信号を調停回路 5 に対して出力し、調停

6

回路 5 はプロセッサ #0 (3) に対してアクノーリッジ信号を返したため、プロセッサ #1 (4) 側で実行される演算命令 31 の IF ステージは、プロセッサ #0

(3) 側の演算命令 30 の IF ステージが完了するまで待たされ、2 クロックサイクル分続いている。同様に、プロセッサ #0 (3) で実行される演算命令 32 も、プロセッサ #1 (4) で実行される演算命令 31 の IF ステージが完了するまで待たされるために 2 クロックサイクル分続き、以下、演算命令 33、34、35 に関しても同様である。次に、演算命令 32 の E ステージにおいては、プロセッサ #0 (3) は調停器 5 からアクノーリッジ信号を受けて演算器 6 を使用しているが、演算処理に 2 クロックサイクル要している。このため、演算命令 32 の E ステージと演算命令 33 の E ステージとが重複することになる。この場合プロセッサ #1 (4) は、演算器 6 へアクセスするためのリクエスト信号を調停回路 5 に対し送出するが、演算器 6 は既にプロセッサ #1 (0) により使用されているためプロセッサ #1 (4) のリクエストは拒絶され、演算命令 32 の E ステージが完了するまで待たされる。このため、プロセッサ #1 (4) の演算命令 33 の E ステージは 2 クロックサイクル分続き、以下、演算命令 34、35 に関しても同様に 2 クロックサイクル続くことになる。

【0012】図 5 は、本実施の形態における 1 チップマルチプロセッサにおいて、2 つのプロセスを実行したときの様子を示したものである。41 は、プロセッサ #0 (3) で、42 は、プロセッサ #1 (4) でそれぞれ実行されているプロセスを示す。また、40 はシングルプロセッサにおいて時分割制御により同様の処理を実行したときの様子を示しており、41、42 との比較のために示している。この実施の形態における 1 チップマルチプロセッサでは、複数のプロセッサが使用するリソースの競合が発生した場合に、対応するパイプラインステージが延びるため、プロセス A (4001)、プロセス B (4002) とともに単体では処理時間が長くなっているが、競合しないとき、あるいは、命令メモリ、データメモリ、演算器を使用しないステージに関しては 2 つのプロセッサで並列に処理を実行することが可能であるため、また、プロセススイッチ 4003 が発生しないため、総合的な処理時間は大幅に短縮されている。また、本発明の調停回路はメモリのみならず、演算器 6 についての調停も行うことができるため、各プロセッサで演算器 6 を共有することができる。この結果、各プロセッサが演算器を内部に搭載するときに比べて大幅に面積を減少させることができる。特に、多数のプロセッサが配置される場合には省スペース化に大きく貢献することになる。

【0013】以上のように、複数のプロセスを実行したときにプロセススイッチが発生せずに効率よく処理を実行することができ、またメモリや演算器を複数のプロセ

ッサで共用するように構成したので、省スペースな1チップマルチプロセッサを実現することができる。

【0014】実施の形態2. 以上の実施の形態1では、複数のプロセッサに共用されているリソース（命令メモリ1、データメモリ2、演算器6）を各プロセッサが使用する際に、調停回路に対してリクエストを出し、調停回路から使用を許可されたプロセッサが、目的のリソースを使用するようにしたものであるが、次に一定の時間間隔でリソースを使用することができるプロセッサを切り替えるようにする場合の実施の形態を示す。この実施の形態では、調停回路5はタイマーカウンタを持ち、ある一定時間間隔毎にリソースを使用することを許可するプロセッサを切り替える。つまり、命令メモリ1、データメモリ2のみならず、演算器6についても、使用するプロセッサを時間ごとに切り替えるものである。図6は、この実施の形態におけるプロセッサと調停回路との信号の送受信の動作を説明するためのブロック図で、7、8は、調停回路5からそれぞれプロセッサ#0

(3)、プロセッサ#1(4)に出力される制御信号であり、9、10は、それぞれプロセッサ#0(3)、プロセッサ#1(4)から調停回路5に出力される制御信号である。制御信号7、8は、それぞれのプロセッサに対して、現在リソースを使用することを許可するか否かを示す信号であり、制御信号9、10は、それぞれのプロセッサが、現在、リソースを使用しているか否かを示す信号である。

【0015】次に動作について説明する。タイマーカウンタがある一定の値に達するとタイマーカウンタはリセットされ再びカウントを開始するとともに、調停回路5は、プロセッサ#0(3)に対して制御信号7をアサートする。プロセッサ#0(3)は、制御信号7がアサートされると、調停回路5に対して、制御信号9をアサートする。このとき、制御信号8、10はネゲートされたままである。さらにタイマーカウンタが再びある一定の値に達するとタイマーカウンタは再びリセットされ、調停回路5は、プロセッサ#0(3)に対する制御信号7をネゲートする。プロセッサ#0(3)は、制御信号7がネゲートされると、現在、処理中の命令を一時停止して、リソースの使用をやめるとともに、制御信号9をネゲートする。調停回路5は、制御信号9がネゲートされると、制御信号8をアサートし、プロセッサ#1(4)に対してリソースを使用することを許可する。プロセッサ#1(4)は、制御信号8がアサートされると、制御信号10をアサートし再び制御信号8がネゲートされるまで、処理を実行する。以下、同様にして、プロセッサ#0とプロセッサ#1が交互にある時間間隔毎に処理を実行していく。図7は、本実施の形態におけるパイプラインを示している。IF、D、E、Wの各ステージ及び30、31、32、33、34、35の各演算命令は実施の形態1の場合と同様である。一定時間ごとにプロ

セッサ#0(3)とプロセッサ#1(4)が交互に共有リソース（命令メモリ1、データメモリ2、演算器6）を使用し、一方のプロセッサが共有リソースを使用している間は、他方のプロセッサは共有リソースを使用していない。図8は、この実施の形態における1チップマルチプロセッサにおいて、2つのプロセスを実行しているときの様子を示したものである。43は、プロセッサ#0(3)で、44は、プロセッサ#1(4)でそれぞれ実行されているプロセスを示す。また、40はシングルプロセッサにおいて時分割制御により同様の処理を実行したときの様子を示しており、41、42との比較のために示している。4004は、一方のプロセッサが動作している間に他方が処理を一時停止している期間である。この実施の形態の場合、2つのプロセス自体を処理するのに要する時間は、単一のプロセッサで実行した場合と同一であるが、プロセススイッチが発生しないため、総合的な処理時間は短くなっている。以上のように、一定時間毎に、動作するプロセッサを切り替えるように構成したため、プロセススイッチが発生せずに、効率よく処理を実行することが可能である。また、各プロセッサからのリソース使用の要求の競合が発生しないため、調停回路、及び、プロセッサのタイミング制御を単純な回路で実現することが可能である。また、一時停止しているプロセッサに対してはクロックの供給を止めるなどして、消費電力を低減させることが可能である。

【0016】

【発明の効果】以上のように、この発明の1チップマルチプロセッサによれば、複数のプロセスを実行したときにプロセススイッチが発生せずに効率よく処理を実行することができ、またメモリのみならず演算器をも複数のプロセッサで共用するように構成したので、省スペースな1チップマルチプロセッサを実現することが可能である。また、メモリのみならず演算器についても、タイマーにより一定時間間隔で、使用するプロセッサを完全に切り替えるので、効率よく処理を実行することができる省スペースな1チップマルチプロセッサを、容易に実現することが可能である。

【図面の簡単な説明】

【図1】 この発明の1チップマルチプロセッサを示すブロック図。

【図2】 この発明の1チップマルチプロセッサを示すブロック図。

【図3】 この発明の1チップマルチプロセッサにおける演算命令のパイプライン図。

【図4】 この発明の1チップマルチプロセッサにおいて、2つのプロセッサで同時に演算命令を実行したときのパイプライン図。

【図5】 この発明の実施の形態1における1チップマルチプロセッサにおいて、2つのプロセスを実行したときの動作図。

【図6】 この発明の実施の形態2における1チップマルチプロセッサにおける調停回路およびプロセッサの構成図。

【図7】 この発明の1チップマルチプロセッサにおいて、2つのプロセッサで同時に演算命令を実行したときのパイプライン図。

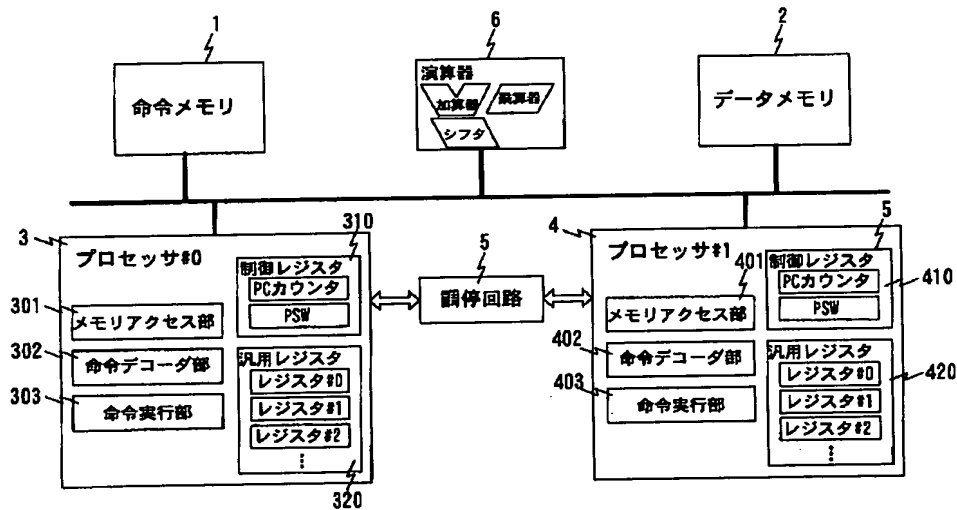
【図8】 この発明の実施の形態2における1チップマルチプロセッサにおいて、2つのプロセスを実行したときの動作図。

【図9】 従来の単一のプロセッサにおいて、2つの

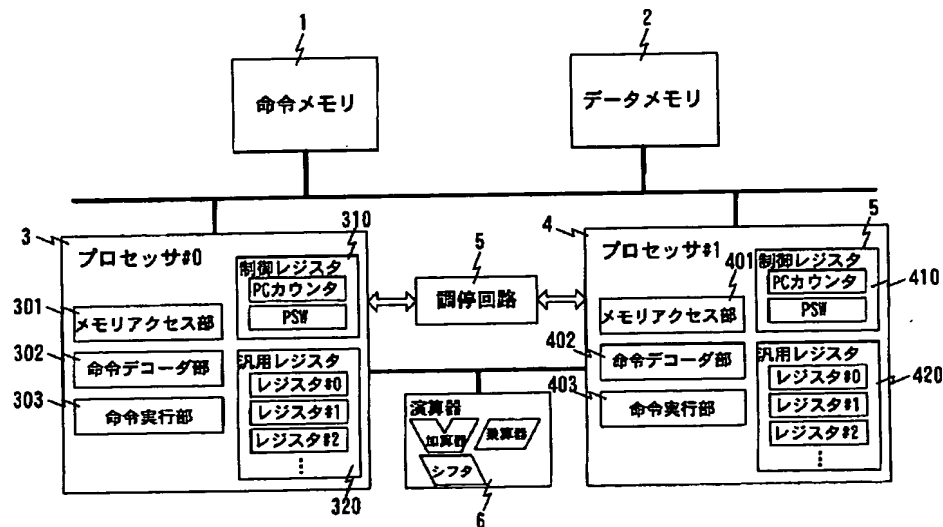
10 プロセスを実行したときの動作図。

【符号の説明】

【図1】

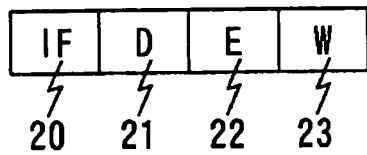


【図2】

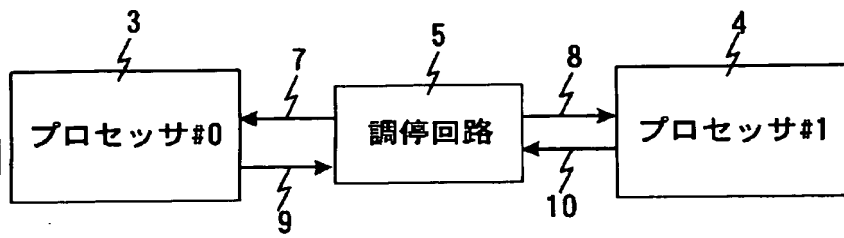


【図3】

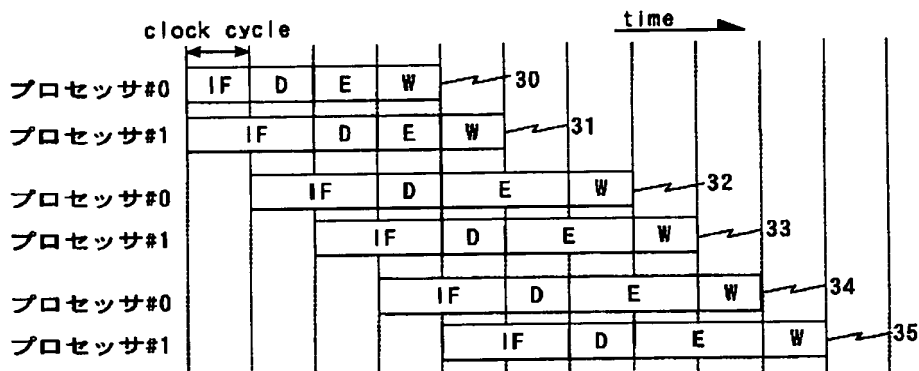
clock cycle



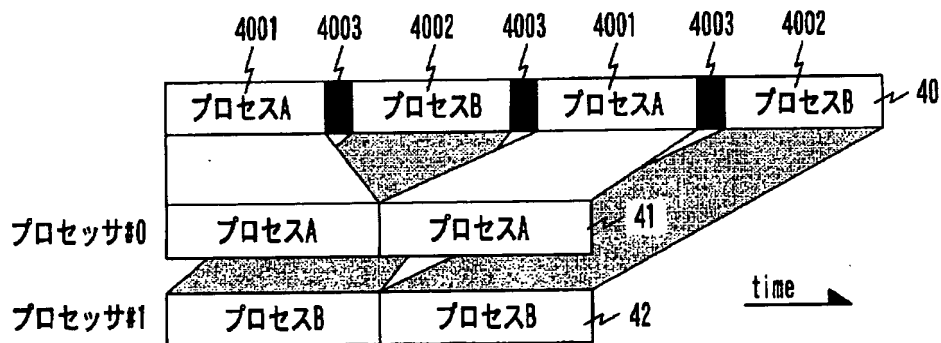
【図6】



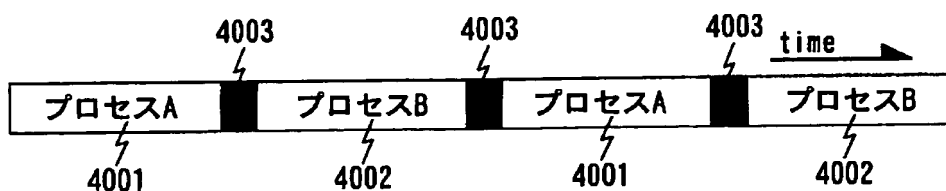
【図4】



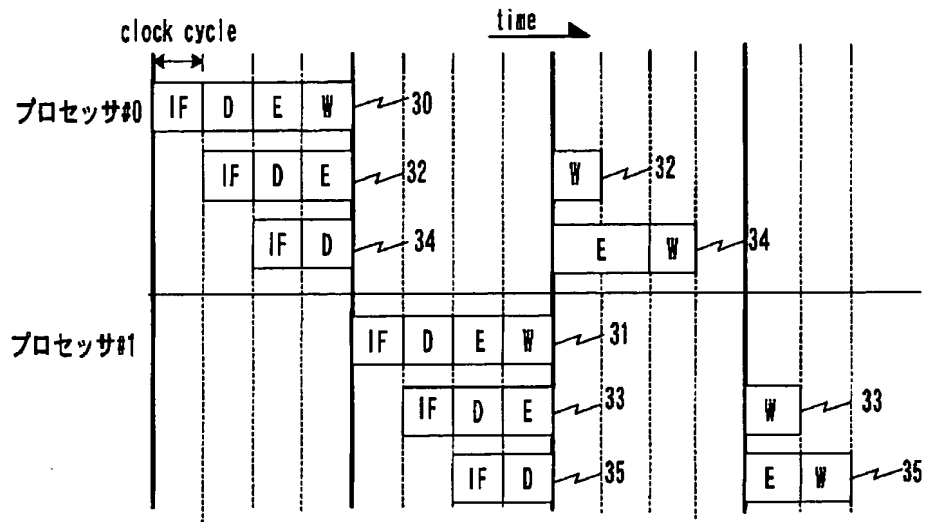
【図5】



【図9】



【図 7】



【図 8】

